Homework #1 Due Friday before class.  
Please try out Gradescope before then!  
(You can submit multiple times, so do a test run on the first homework.)

Office hours now posted on the web page:

<table>
<thead>
<tr>
<th>Name</th>
<th>Day/Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evan McCarty</td>
<td>Thu, 3-4pm</td>
<td>CSE 031</td>
</tr>
<tr>
<td>Merit Saglam</td>
<td>Tue, 11-1pm</td>
<td>CSE 031</td>
</tr>
<tr>
<td>Kostas Kalivas</td>
<td>Thu, 10:30-11:30am</td>
<td>CSE 213</td>
</tr>
<tr>
<td>Eren Hanches</td>
<td>Wed, 3-4pm</td>
<td>CSE 031</td>
</tr>
<tr>
<td>Ian Turner</td>
<td>Wed, 12-1pm</td>
<td>CSE 218</td>
</tr>
<tr>
<td>Junhao (Ian) Zhu</td>
<td>Thu 4-5pm</td>
<td>CSE 031</td>
</tr>
</tbody>
</table>

Class e-mail list, Discussion board

---

Sessions start this week:

<table>
<thead>
<tr>
<th>Section</th>
<th>Day/Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA Evan</td>
<td>Th, 12:30-1:30pm</td>
<td>THO 234</td>
</tr>
<tr>
<td>AB Mert</td>
<td>Th, 1:30-2:20pm</td>
<td>DEN 217</td>
</tr>
<tr>
<td>AC Ian</td>
<td>Th, 2:30-3:20pm</td>
<td>MGH 254</td>
</tr>
<tr>
<td>AD Krista</td>
<td>Th, 11:30-12:20pm</td>
<td>MGH 251</td>
</tr>
</tbody>
</table>

---

a combinatorial logic example

We would like to compute the number of lectures or quiz sections remaining at the start of a given day of the week.

- Inputs: Day of the Week, Lecture/Section flag
- Output: Number of sessions left

Examples: 
- Input: (Wednesday, Lecture)  Output: 2
- Input: (Monday, Section)  Output: 1

Sessions of class:

---

implementation in software

public int classesLeft (weekday, lecture_flag) {
    switch (day) {
        case SUNDAY:
            return lecture_flag ? 3 : 1;
        case MONDAY:
            return lecture_flag ? 3 : 1;
        case TUESDAY:
            return lecture_flag ? 2 : 1;
        case WEDNESDAY:
            return lecture_flag ? 2 : 1;
        case THURSDAY:
            return lecture_flag ? 1 : 1;
        case FRIDAY:
            return lecture_flag ? 1 : 0;
        case SATURDAY:
            return lecture_flag ? 0 : 0;
    }
}

---

implementation with combinational logic

Encoding:
- How many bits for each input/output?
- Binary number for weekday
- One bit for each possible output

Weekday
- 0 = Sunday
- 1 = Monday
- 2 = Tuesday
- 3 = Wednesday
- 4 = Thursday
- 5 = Friday
- 6 = Saturday

Lecture?
- 0 = Lecture
- 1 = Section

---

defining our inputs

public int classesLeft (weekday, lecture_flag) {
    switch (day) {
        case SUNDAY:
            return lecture_flag ? 3 : 1;
        case MONDAY:
            return lecture_flag ? 3 : 1;
        case TUESDAY:
            return lecture_flag ? 2 : 1;
        case WEDNESDAY:
            return lecture_flag ? 2 : 1;
        case THURSDAY:
            return lecture_flag ? 1 : 1;
        case FRIDAY:
            return lecture_flag ? 1 : 0;
        case SATURDAY:
            return lecture_flag ? 0 : 0;
    }
}

---

converting to a truth table

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Number</th>
<th>Binary</th>
<th>Lecture?</th>
<th>c0</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>0</td>
<td>[000]</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Monday</td>
<td>1</td>
<td>[001]</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Tuesday</td>
<td>2</td>
<td>[010]</td>
<td>001</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wednesday</td>
<td>3</td>
<td>[011]</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Thursday</td>
<td>4</td>
<td>[100]</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
<td>[101]</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Saturday</td>
<td>6</td>
<td>[110]</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### logic (part one)

- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

### Boolean algebra

- Binary operations: $\{+, \cdot\}$
- Set $B$ contains at least two elements: $0, 1$
- A set of elements $B$ containing $\{0, 1\}$ such that the following axioms hold:
  1. Closure:
     - $a \cdot b$ is in $B$
     - $a + b$ is in $B$
  2. Comutativity:
     - $a + b = b + a$
     - $a \cdot b = b \cdot a$
  3. Associativity:
     - $(a + b) + c = a + (b + c)$
     - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  4. Distributivity:
     - $(a + b) \cdot (a + c) = a \cdot b + a \cdot c$
  5. Complementarity:
     - $a + a' = 1$

### gates

- Multiple input AND gates
- (multiple input AND gates)

### logic (part two)

- $c_3 = (d_2 = \text{MON and LEC})$ or $(d_1 = \text{TUE and LEC})$

### gates

- D2
- D1
- D0

### logic to gates

- $c_3 = d_2 \cdot d_1 \cdot d_0 \cdot L + d_2 \cdot d_1 \cdot d_0 \cdot L$

### Boolean algebra

- $a + (b + c) = (a + b) + c$
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- $a \cdot 1 = a$
- $a + 0 = a$

### gates

- D2
- D1
- D0

### logic to gates

- (multiple input AND gates)

### logic (part three)

- $c_3 = d_2 \cdot d_1 \cdot d_0 \cdot L + d_2 \cdot d_1 \cdot d_0 \cdot L$

### gates

- D2
- D1
- D0

### logic to gates

- (multiple input AND gates)
axioms and theorems of Boolean algebra

<table>
<thead>
<tr>
<th>Identity</th>
<th>Null</th>
<th>Idempotency</th>
<th>Involution</th>
<th>Complementarity</th>
<th>Commutativity</th>
<th>Associativity</th>
<th>Distributivity</th>
<th>Absorption</th>
<th>Factoring</th>
<th>Consensus</th>
<th>De Morgan's</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( X + 0 = X )</td>
<td>2. ( X + 1 = 1 )</td>
<td>3. ( X + X = X )</td>
<td>4. ( (X')' = X )</td>
<td>5. ( X + X' = 1 )</td>
<td>6. ( X + Y = Y + X )</td>
<td>7. ( (X + Y) + Z = X + (Y + Z) )</td>
<td>8. ( X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z) )</td>
<td>9. ( X \cdot Y + X \cdot Y' = X )</td>
<td>10. ( (X + Y) \cdot (X + Y') = X )</td>
<td>11. ( (X + Y) \cdot Y = X \cdot Y )</td>
<td>12. ( (X + Y) \cdot (X + Z) = X \cdot Z + X' \cdot Y )</td>
</tr>
</tbody>
</table>

Identity:
1. \( X + 0 = X \)

Null:
2. \( X + 1 = 1 \)

Idempotency:
3. \( X + X = X \)

Involution:
4. \( (X')' = X \)

Complementarity:
5. \( X + X' = 1 \)

Commutativity:
6. \( X + Y = Y + X \)

Associativity:
7. \( (X + Y) + Z = X + (Y + Z) \)

Distributivity:
8. \( X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z) \)

Absorption:
9. \( X \cdot Y + X \cdot Y' = X \)

Factoring:
10. \( (X + Y) \cdot (X + Y') = X \)

Consensus:
11. \( (X + Y) \cdot Y = X \cdot Y \)

De Morgan's:
12. \( (X + Y + \ldots)' = X' \cdot Y' \cdot \ldots \)

axioms and theorems of Boolean algebra

proving theorems (rewriting)
Using the laws of Boolean Algebra:
Prove the theorem:
\[
X \cdot Y + X \cdot Y' = X
\]
Distributivity (8)
Complementarity (5)
Identity (1D)
Prove the theorem:
\[
X + X \cdot Y = X
\]
Identity (1D)
Distributivity (8)
Identity (1D)

proving theorems (truth table)
Using complete truth table:
For example, de Morgan's Law:
\[
(X + Y)' = X' \cdot Y'
\]
NOR is equivalent to AND with inputs complemented
\[
\begin{array}{ccc|c|c}
X & Y & X' & Y' & X \cdot Y' \\
\hline
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

simplifying using Boolean algebra
\[
c_3 = \overline{d_2} \cdot \overline{d_1} \cdot d_0 \cdot L + \overline{d_2} \cdot d_1 \cdot \overline{d_0} \cdot L
\]
\[
\overset{1}{=} \overline{d_2} \cdot \overline{d_1} + \overline{d_0} \cdot L
\]
\[
\overset{2}{=} \overline{d_2} \cdot \overline{d_1} \cdot (d_0 + \overline{d_0}) \cdot L
\]
\[
\overset{3}{=} \overline{d_2} \cdot \overline{d_1} \cdot L
\]

simplifying using Boolean algebra
\[
c_3 = \overline{d_2} \cdot \overline{d_1} \cdot d_0 \cdot L + \overline{d_2} \cdot d_1 \cdot \overline{d_0} \cdot L
\]
\[
\overset{1}{=} \overline{d_2} \cdot \overline{d_1} + \overline{d_0} \cdot L
\]
\[
\overset{2}{=} \overline{d_2} \cdot \overline{d_1} \cdot (d_0 + \overline{d_0}) \cdot L
\]
\[
\overset{3}{=} \overline{d_2} \cdot \overline{d_1} \cdot L
\]
1-bit binary adder

- **Inputs:** A, B, Carry-in
- **Outputs:** Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ S = A' B' \text{Cin} + A' B \text{Cin}' + A B' \text{Cin} + A B \text{Cin}' + A B \]

\[ \text{Cout} = A' B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + A B \]

The theorems of Boolean algebra can simplify expressions—e.g., full adder's carry-out function

- NOT
  \[ X' \quad \overline{X} \quad \neg X \]

- AND
  \[ X \cdot Y \quad XY \quad X \land Y \]

- OR
  \[ X + Y \quad X \lor Y \]

apply theorems to simplify expressions

more gates

NAND
\[ \neg(X \land Y) \quad (XY)' \]

NOR
\[ \neg(X \lor Y) \quad (X + Y)' \]

XOR
\[ X \oplus Y \]

XNOR
\[ X \leftrightarrow Y \]

a 2-bit ripple-carry adder
mapping truth tables to logic gates

Given a truth table:
1. Write the Boolean expression
2. Minimize the Boolean expression
3. Draw as gates
4. Map to available gates

\[
F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} + ABC
\]
\[
= \overline{A}(B + C) + A\overline{B}(\overline{C} + C)
\]
\[
= \overline{A}(B + C) + AC
\]

canonical forms

- Truth table is the unique signature of a Boolean function
- The same truth table can have many gate realizations
  - we've seen this already
  - depends on how good we are at Boolean simplification
- Canonical forms
  - standard forms for a Boolean expression
  - we all come up with the same expression

sum-of-products canonical form

- also known as Disjunctive Normal Form (DNF)
- also known as minterm expansion

Product term (or minterm)
- ANDed product of literals
- input combination for which output is true
- each variable appears exactly once, true or inverted (but not both)

F in canonical form:
\[
F(A, B, C) = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} + ABC + ABC'
\]
\[
= (\overline{A} + A)(B' + B)(\overline{C} + C)
\]
\[
= C + ABC'
\]
\[
= AB + C
\]

product-of-sums canonical form

- Also known as Conjunctive Normal Form (CNF)
- Also known as maxterm expansion

Complement of function in sum-of-products form:
- \( F' = \overline{A}'\overline{B}'C + \overline{A}'BC' + ABC' \)

Complement again and apply de Morgan’s and get the product-of-sums form:
- \( (F')' = (\overline{A}'\overline{B}'C + \overline{A}'BC' + ABC')' \)
- \( F = (A + B + C) (A' + B' + C) \)
product-of-sums canonical form

Sum term (or maxterm)
- ORed sum of literals – input combination for which output is false
- each variable appears exactly once, true or inverted (but not both)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>maxterms</th>
<th>F in canonical form:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( A + B + C )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( A + B + C' )</td>
<td>canonical form = minimal form</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( A + B + C' )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( A + B' + C )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( A + B' + C )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( A + B' + C )</td>
<td>canonical form = minimal form</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( A + B' + C )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( A + B + C )</td>
<td>( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) )</td>
</tr>
</tbody>
</table>

\( F(A, B, C) = (A + B + C)(A + B' + C)(A' + B + C) \)

\( F(A, B, C) = (A + B + C)(A + B' + C) \)

\( F(A, B, C) = (A + B + C)(A' + B + C) \)

\( F(A, B, C) = (A + C)(B + C) \)